A Comparative Study of S-shape and Concave Software Reliability Growth Models

Ela Kashyap[1]
Amity School of Engineering and Technology
Amity University
Noida, Uttar Pradesh,
ekashyap@amity.edu

Ajay Rana[2]
Amity School of Engineering
Amity University
Noida, Uttar Pradesh,
ajay_rana@amity.edu

Abstract— **The requirements of software users are changing dynamically with a rapid pace which is attributed to the competitive environment and strong market position. The software industry needs to accomplish wearisome task of producing and upgrading the quality application software cost-effectively. With the advent of new technology, the upgradations and enhancements in the software have become more viable. Equipped with technology, the software industry releases new software rapidly at regular time intervals. Apart from the quick releases, the utmost need of the hour is to ensure that the software will perform without failure for specific duration of time under specified conditions to meet the expectations of the users. Therefore, the prediction and quantification of software reliability is consequential. Software reliability growth models (SRGM) are robust tools used for the quantitative prediction of software reliability. SRGM use failure data to graphically analyze the pattern of failure detection rate to measure software reliability, in terms of two basic curves,- S-shape curve and Concave curve. The focus of the paper is to review the basic SRGMs on the basis different shapes of the failure curve attained during the software reliability quantification process.**

Keywords -Software reliability growth model, S-shape growth curve, Concave shape curve, software reliability

I. INTRODUCTION

In this era of technological advancements the computers and software have become an integral part of our lives. With the involvement of software in every sphere of human life, the utility of software has increased. The application software have marked their presence in all spans of life from mission critical applications like defense, military to safety application such as medical procedures [10]. The failure of such software may result in financial setback, loss of life or failure of a critical operation. Therefore, these vital applications require reliable software to perform relentlessly without failure [2]. Hence, prior to releasing the software product it is very important to determine whether it will perform as per the user expectation without failure [10,11]. Moreover, with the growth of existing market, a constant increase in the size and complexity of the software has been observed. In this scenario, the quantification of software reliability is the utmost need of the hour. The reliability is expressed in terms of failure free deliberation of an anticipated function, for a specified duration of time in a given operational environment [3][9]. Hence taking this into consideration measuring software reliability is very desirable to quantitatively analyze the software reliability. SRGMs are mathematical models that statistically analyze software reliability, hence confer an account of the quality of software product. SRGM quantifies software reliability by establishing the mathematical relationships between testing time and the rate at which failures occur at the time of testing. These relationships are established in terms of stochastic processes, probability and statistical formulas. In the testing phase, the failure data is gathered which is used in SRGM to predict the software reliability during its operational tenure in the future [11][16][14]. The failure data is considered as input to the SRGM which produces the reliability prediction as the output in the terms of mathematical functions such as exponential or logarithmic functions. These functions are accountable for the prediction of time between the failure[14]. When the failure data is plotted in terms of 'Cumulative usage time' and 'cumulative number of faults detected', the two basic shapes are observed known as S-shape and concave [16].

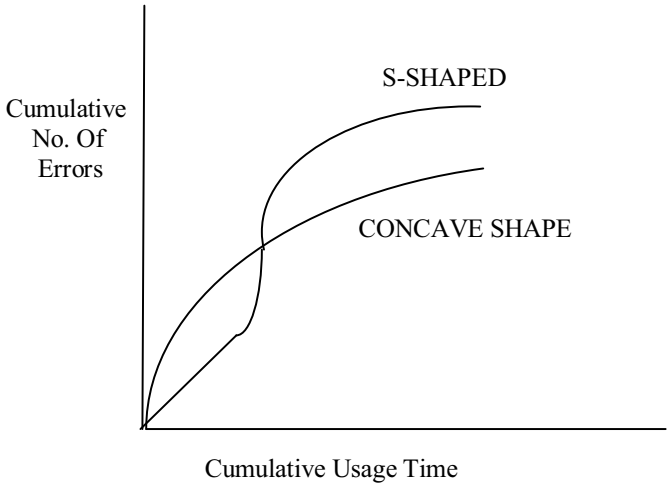

Figure 1. Concave and S-shape obtained by software reliability growth model.

978-1-5090-0076-0/15 $31.00 © 2015 IEEE
DOI 10.1109/CICN.2015.280

Both the s-shape and concave curve depict the asymptotic behavior i.e. a finite asymptotic value is attained by both the curves because the fault rate plunges down steadily as the defects are detected and repaired during the tenure of testing [2].

II. CONCAVE MODELS

The Concave shaped models are Decreasing Failure Rate (DFR) models. In these models the failure rate decreases at a constant pace as the number of faults are detected and removed. The idea behind DFR is that as the given pre-determined number of errors are detected and removed, the software reliability improves [4]. In these models when failure data is supplied as input, the failure rate reduces steadily and becomes constant after some time, during the testing tenure. The constant decrease in the failure rate is attributed to regular detection and removal of the faults at a constant pace during the course of testing. Hence, the concave shape pattern is obtained [2]. Goel-Okumoto, Musa and Jelinksi-Moranda models are amongst the earliest concave shape models. The paper describes these models in detail in the following section.

A. Jelinski Moranda Model

Jelinski and Moranda model (J-M Model) is one of the primitive software reliability models. It is based upon the Markov process, according to which the failure is represented by countable states. These states are expressed in terms of a hazard function. The model established a relationship between the hazard rate and number of faults corrected or remaining in the software. It states that the hazard rate constantly varies each time when a fault is corrected which means that the hazard rate remains constant between failures but decreases gradually as faults are removed [17]. This is the reason why the hazard rate versus time graph depicts a step pattern.

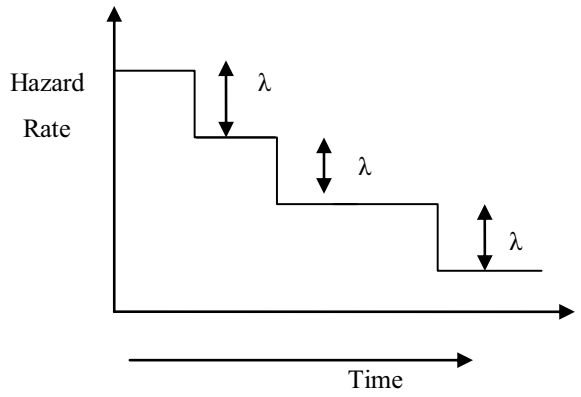

Figure 2. Relationship between Hazard rate and time in J-M model

The J-M model made the following assumptions:

- The failure rate is constant and proportional to the current fault content remaining in the software [15].
- The initial error content is also constant and remains unpredictable [15].
- The failure which may occur due to a fault or error is random in nature and occurs according to an exponential distributed function [17].
- The time between the failures i and i-1 are independent of each other [1].

The following mathematical expression gives the hazard function or failure rate calculated between i^{th} and $i\text{-}1^{th}$ failure:

$$Z(ti)=\phi(N-(i-1)) \qquad i=1,2,3,4,....N \text{ [12]}$$

ϕ= Proportionality constant determining the failure rate per fault

N= The count of initial number of faults

ti= Time elapsed between the i^{th} and $i\text{-}1^{th}$ failures

This process of determining the fault detection rate on the basis of number of fault content remaining in the system is known as "De-eutrophication" process [7]. In this model, since the hazard rate constantly decreases, hence the failure detection rate increases steady and becomes constant the shape of the curve is concave.

B. MUSA basic execution model

MUSA model is pioneer in determining the software reliability in terms of execution time which is expressed as the time duration for which the processor was actually utilized. In this model a relationship between the execution time and failure rate was established. The model defined execution time as the measure of processor utilization during the execution of the program [7][8].The model inferred that the software reliability can be quantified more precisely in terms of the execution time as compared to the other measures of software reliability [12]. Moreover, it states that the analysis and measurement of failure rate can be done more effectively when time of execution of the software is more. The number of failures occurring in a given time interval is dependent upon the execution history of the software. It is a perfect debugging model in which when the fault is removed then it is considered to be removed perfectly and it will not lead to the introduction of the new faults [8].

The model made the following assumptions [7][8]:

- When a failure is encountered, the immediate action is taken to eradicate the fault which is the root cause of failure.
- The failure intensity plunges down exponentially with the increase in the number of failures detected.
- All of the failures are taken into consideration.
- The failure intensity decreases steadily with a constant rate during the testing.

- The execution time is expressed in terms of Non-Homogeneous Poisson Process.

The model expresses the Mean Value function as follows:

$m(t)=a(1- e-(\lambda/a)t)$ [12]

λ = Failure intensity

m(t) = Expected number of failure experienced in time duration 't'

a = The total number of faults occur in infinite time

t = Execution time

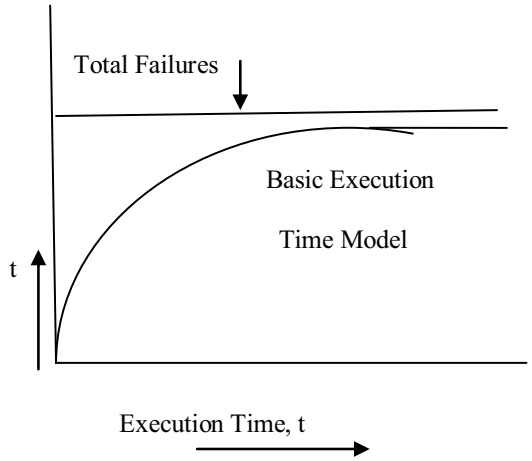

Figure 3. Concave shape Musa Basic execution model

During the course of testing the failure intensity decreases constantly as the decreasing function of the execution rate. Hence, the number of faults detected showed a steady increase and gradually became constant which is accountable for the constant rise in the failure detection rate. Therefore, the curve attained a concave shape.

C. Goel and Okumoto

Before Goel and Okumoto (G-O) model breaks new grounds in the area of software reliability modeling based on NHPP. This exponential SGRM is a time dependent failure rate detection model [5].G-O model is an imperfect debugging model in which the removal of one fault may lead to the introduction of the other faults.

The model made the following assumptions [1][12]:

- The failure rate is dependent upon the count of the faults remaining in the software
- The Non-Homogeneous Poisson Process form the bases of failure detection
- When a failure is detected, the fault that leads to the failure is determined and eradicated before further testing.
- The time elapsed between two consecutive failures is proportional to the time to failure.
- The debugging process is considered to be perfect in which no new fault will be introduced

The hazard function is mathematically expressed as follows:

$z(\tau)=\phi f(N-nc$ [12]

$z(\tau)$=hazard function

τ= execution time

f= linear execution frequency

ϕ= proportionality constant

nc= Number of the faults corrected between interval $\{0, \tau\}$

The Mean Value Function is described as follows:

$m(t)=a(1-e^{-bt})$ [1]

m(t)= Mean value function

b(t)= Rate of fault detection

a(t)= Error content function

G-O model is different from Jelinski & Moranda model, in terms of assumptions made with regard to the initial error content .In G-O model the initial error content is taken as a random variable, which can't be a pre-determined fixed constant as proposed in the J-M model. The J-M model suggested that the time between failures is independent whereas the G-O model assumes that the time duration between two concurrent failures i and i-1 is dependent on the time to failure i-1 [1][12]. This model is an imperfect debugging model.

III. S- SHAPED MODEL

The models depicting S-shape patterns also demonstrate the asymptotic behavior similar to the concave model. The failure data which is used to plot the curve is analyzed in two phases of software testing. In the early phase, the testing is comparatively less effective than the later phase because the testing team performs testing using same test cases as used by the development team, therefore the failure rate decreases. This is the reason why the curve attains the inward bulge. Later on, in the application testing phase, the new defects are uncovered. Therefore in the later stages of testing, the S-shape curve behaves in the same manner like the concave–curve [2]. The next section illustrates the popular s-shape models.

A. Yamada Weibull Effect Model

The Weibull model was proposed by Yamada that established the relationship between testing effort and current error content. The testing effort is the effort spent in terms of resources consumed during testing in order to detect and remove errors during arbitrary time tenure.

The model made the following assumptions [18]:

- The system failures occur at random number of times due to the faults present in the software.
- The initial error content is expressed in terms of a random variable.
- The time elapsed between two consecutive failures is proportional to the time to failure.
- Whenever the failure is encountered, the actions are taken to remove the cause of failure immediately.

In this model the following equation for Mean Value Function is stated as:

$m(t)=a(1-\exp(-r\ W(t)))$ [18]

m(t)= Mean Value Function
a= Mean value of initial Error content
W(t)= testing effort
r= rate of error detection

The testing effort reduces during the tenure of testing since the number of resources consumed decreases as the testing proceeds. Hence the testing effort depends upon the availability of the testing resources consumed [13].

B. Pham and Nordmann (1997b)

Pham model is a generalized S-shape imperfect debugging NHPP software reliability model that provides a mathematical expression for the estimation of software reliability based upon the calculation of Mean value function (MVF). According to this model the calculation of software reliability is divided into three steps procedure. The first step involves stochastically calculating the cumulative numbers of errors during a time span 't'. In next step, the mean value function is defined in terms of error content function 'a(t)' and Error detection rate 'b(t)'. Finally the model is analyzed using test data and Mean value function is calculated. MVF is measure of the number of faults accumulated in the software in any given time 't' [6].

The model made the following assumptions:

- Each fault has different error detection rate.
- The detection of a failure leads to the removal of error causing it but it may introduce new fault.

The equation of Mean Value Function is calculated as below:

$$m(t)=e^{-B(t)}\left[m_o + \int_{t_o}^{t} a(t)b(t)\, e^{B(t)} dt \right] \qquad [12]$$

$$B(t)= \int_{t_o}^{t} b(t)\, dt \qquad [12]$$

m(t)= Mean value function
b(t)= fault detection rate
a(t)= error content function
initially $m(t_0)=m_0$, t_0 is the time to begin debugging process

The error content function a(t) predicted a imperfect debugging trend because the total numbers of errors augmented at a steady rate because the eradication of errors lead to the introduction of new errors in the system. It was observed that the fault detection rate b(t) surged up as well because the software tester expertise in error detection improves with the course of testing process [6].

C. KG Model

The model focused upon the eradication of additional faults in the process of removal of the intended faults detected during the testing process, assuming that the removal of additional faults not lead to any failure conditions. The model categorized the faults as dependent and independent faults. The independent faults are those faults which are initially detected and eradicated during the course of testing process. The dependent faults are those which are removed additionally during the eradication of independent faults [13,12].

The model considered the following assumptions [13]:

- The remaining error-content may lead to the software failure at random number of times.
- While detecting and removing the existing faults during the error detection process, the tester may encounter some additional errors.
- The removal of the existing error will not introduce the new errors.
- The complete duration of a software development life cycle often exceeds the optimal release time.
- There is an error-detection procedure followed at the manufacturer's end, then subsequent procedure must also be adopted at the user's end.

The mean value function is calculated as follows:

mr(t)= a[(1-e-(p+q)t)/ 1+(q/p) e-(p+q)t] [12]
mr(t)= Mean Value function
p=failure detection rate for leading faults
q=failure detection rate for dependent faults
a= Error content

The model illustrates that the s-shaped growth curve is obtained by virtue of fault dependency and debugging time lag whereas the failure phenomenon is expressed in terms of exponential growth curve [12][13].

IV. CONCLUSIONS

The failure data observed during the time of testing can be utilized in the quantification of software reliability. Hence the time to failure can be predicted and software testers may decide when to stop testing and release software. In concave shape models a gradual dip in the failure rate occurs when errors are detected and removed during testing. Jelinski Moranda and Musa model are simple models based upon the concept of perfect debugging where a direct relationship existed between the failure rate and number of faults generated. The perfect debugging models assume that the detection of faults will not introduce the new faults and the error content is reduced as the faults are removed.Musa model depicted that the software reliability can be quantified more precisely in terms of execution time as compared to the other measures of software reliability. G-O model explained that the initial fault content must be a random variable and the time between two failures is dependent upon the time to failure. The statistical interpolation of the s-shape graphs depicted that the failure detection rate initially reduces because at the time of testing

the testing team performs the same test cases as created by the development team but at the later stages as it introduces the new test cases, the failure rate surges up justifying the s-shapedness of the failure curve. These models are imperfect debugging models since the introduction of new faults in the process of failure removal may lead to additional faults, hence fault removal rate is not constant but it varies during the course of testing.

TABLE I. COMPARISON OF SOFTWARE RELIABILITY GROWTH MODEL

Comparison of Software Reliability Growth Models			
Model name	***Model type***	***Nature of Debugging***	***Model Considerations***
Jelinski Moranda J-M Model	Concave	Perfect Debugging Model	Initial Fault content is constant Time between failures is independent
MUSA basic execution model	Concave	Perfect Debugging model	Failure rate is dependent upon the execution time Initial fault content is constant
Goel and Okumoto (G-O) model	Concave	Imperfect Debugging model	Fault content is a random variable Time between two failure is proportional to the time between two failures Removal of fault may lead to the introduction of additional fault
Yamada Weibull Effect Model	S-shape	Imperfect Debugging model	Testing effort is proportional to the current error content. Initial fault content is a random variable
Pham and Nordmann	S-shape	Imperfect Debugging model	Initial fault content is a random variable Fault content may increase due to introduction of new faults.
K-G Model	S-shape, Exponential growth	Imperfect Debugging model	Initial fault content is a random variable Fault content may lead to random number of failures Additional faults are removed in the process of removal of detected faults

REFERENCES

[1] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Trans. Reliab., vol. R-28, no. 3, pp. 206–211, 1979.

[2] A. Wood, "Software reliability growth models," 1996.

[3] B.M.Kumar, "An Empirical Analysis of Software Reliability Prediction through Reliability Growth Model Using Computational Intelligence," Comput. Intell. Data Min., vol. 2, pp. 513–524, 2015.

[4] D. S. Bai and W. Y. Yun, "Optimum Number of Errors Corrected Before Releasing a Software System.," *IEEE Trans. Reliab.*, vol. 37, no. 1, pp. 41–44, 1988.

[5] G. Aggarwal and V.K. Gupta., "Software Reliability Growth models," Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 4, 2014.

[6] H. Pham, System Software Reliability. Verlang, London: Springer, 2011

[7] J. D. Musa, "A theory of software reliability and its application (review of existing methods)," *Softw. Eng. IEEE Trans.*, 1975.

[8] J. D. Musa and K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement," *ICSEproceedings 7th Int. Conf. Softw. Eng.*, pp. 230–238, 1984.

[9] K. Taehyoun, K. Lee, and J. Baik, "An effective approach to estimating the parameters of software reliability growth models using a real-valued genetic algorithm," *J. Syst. Softw.*, vol. 102, pp. 134–144, 2015.

[10] P. K. Kapur., A.Tondon, and G.Kaur, "Multi upgradation software reliability model," in *2nd International Conference on Relaibility, Safety and Hazard(ICRESH)*, 2010, pp. 468–474.

[11] P. K. Kapur, D. N. Goswami, A. Bardhan, and O. Singh, "Flexible software reliability growth model with testing effort dependent learning process," *Appl. Math. Model.*, vol. 32, no. 7, pp. 1298–1307, 2008.

[12] P. K. Kapur, H. Pham, A. Gupta, and P. C. Jha, *Software Reliability Assessment with OR Applications*. London: Springer, 2011,ch.3, pp. 117-151.

[13] P. K. Kapur and R.B Garg., "A software reliability growth model for an error removal phenomenon," *Softw. Eng. J.*, no. 7, pp. 291–294, 1992.

[14] R. Lai and M. Garg, "A detailed study of NHPP software reliability models," *J. Softw.*, vol. 7, no. 6, pp. 1296–1306, 2012.

[15] S.Lohmor and B.B Sagar, "Overview: Software Reliability Growth Model," *Int. J. Comput. Sci. Inf. Technol.*, 2014.

[16] S. Mohamad, S. Mashita, and McBride T., "A comparison of the reliability growth of open source and in-house software," in *15th IEEE Software Engineering Conference (APSEC'08)*, 2008.

[17] S. Peter, "Parameter estimation for a specific software reliability model," *IEEE Trans.*, pp. 323–328, 1985.

[18] S. Yamada, "s-Shaped Software Reliability Growth Models and Their Applications," *IEEE Trans. Reliab. Reliab.*, no. 4, pp. 289–292, 1984.